

Sarbjot Mann
CMPT 125, Assignment 3

1. The Fibonacci sequence is given by 1, 1, 2, 3, 5, 8, 13, 21, The first two numbers are 1, and subsequent numbers are the sum of the previous two. If $\text{Fib}(k)$ represents the k th number in the sequence, then we have $\text{Fib}(1) = \text{Fib}(2) = 1$, and $\text{Fib}(k) = \text{Fib}(k-1) + \text{Fib}(k-2)$ for $k > 2$.

- a. Given the code segment, the call stack should look as follows, if the code is as follows:

```
int Fib(int k) {
    if (k <= 2) {
        return 1;
    }
    return Fib(k-1) + Fib(k-2);
}

int main() {
    int m = 4;
    int n = Fib(m);
    return 0;
}
```

The code will run as follows:

```
fib(4)
  fib(3)
    fib(2)
      fib(1)
    fib(2)
```

The first $\text{fib}(4)$ will run but then it'll call $\text{fib}(3)$ before $\text{fib}(2)$ so after $\text{fib}(3)$ is finished running (it'll run $\text{fib}(2)$ and $\text{fib}(1)$) then $\text{fib}(2)$ which is from the original $\text{fib}(4)$ will run.

Call Stack	Last Func Call if app,	Last Func Return, if app	Function parametres	Func return value if app
Main m = 4 n = ?	Main	N/A	N/A	N/A
Fib k = 4 Main m = 4 n = ?	Fib	N/A	k = 4	N/A
Fib k = 3 Fib k = 4 Main m = 4 n = ?	Fib	Fib(4)	k=3	N/A
Fib k = 2 Fib k = 4 Main m = 4 n = ?	Fib	Fib(3)	k=2	1
Fib k = 1 Fib k = 4 Main m = 4 n = ?	Fib	Fib(3)	k=1	1
Fib k = 2 Main m = 4 n = ?	N/A	Fib(4)	2	1
Main m = 4 n = 3	N/A	Fib	N/A	3

b.

```
#include <stdio.h>

////////////////////////////////////
                /*Question 1b*/
////////////////////////////////////

int FibIter(int k) {
    int n1 = 1;
    int n2 = 1;
    int n3 = 1; //if the user doesn't enter <= 2, return n3 as beign 1
    for (int i = 2; i < k; i++){
        n3 = n1+n2; //adding two previous numbers
        n1 = n2; //set n1 to n2
        n2 = n3; //set n2 to n3
    }
    return n3;
}
```

2.

```
////////////////////////////////////
                /*Question 2a*/
////////////////////////////////////

int findMinpos(int arr[], int mid, int len){
    /*Set up temp index because we keep on comparing mid and mid+1 we won't "Save the value" and will have the last min value instead
    of the least value which may not be the last min value*/

    static int index; //let index be a static int so it is intialized once and doesn't lose its value :)
    static int temp;

    if (temp == 0){
        index = mid; //to start off index to be equal to mid so we have a val to compare to
        temp = temp + 1; //set temp to + 1 so we dont enter here again
    }

    if (mid == len){ //Remember that in a array we count from 0
        temp = 0; //reset temp counter since we may need this after
        return index;
    }
    else if (arr[index] <= arr[mid]){
        findMinpos(arr, mid+1, len); //if index is <= mid, dont change index
    }
    else if (arr[index] > arr[mid]){
        index = mid; //if val of [index] is greater than mid, set index to mid
        findMinpos(arr, mid+1, len);
    }

    return index;
}
```

```
Assignment3.c No Selection
46     else if(arr[index] > arr[mid]){
47         index = mid; //if val of [index] is greater than mid, set index to mid
48         findMinpos(arr, mid+1, len);
49     }
50
51     return index;
52 }
53
54 ///////////////////////////////////////////////////
55 /*Question 2b*/
56 ///////////////////////////////////////////////////
57
58 void swapMinpos(int arr[], int mid, int len) {
59     // Swaps minimum of subarray arr[mid...len-1] with arr[mid]
60     //We just coded this function! Let's use it!
61     int minindex = findMinpos(arr, mid, len); //Find minindex
62     int tmp = arr[minindex]; //Set tmp to arr[minindex]
63     arr[minindex] = arr[mid]; //Assigning mid value to minindex
64     arr[mid] = tmp; //assigning minindex to tmp
65 }
66
67 ///////////////////////////////////////////////////
68 /*Question 2c*/
69 ///////////////////////////////////////////////////
70
71 void selSortR(int arr[], int mid, int length){
72     static int tmp = 0;
73     if (tmp == mid){ //If we are at mid stop
74         return;
75     }
76     else{
77         swapMinpos(arr, tmp, length); //Call out swap function
78         tmp = tmp+1; //Increase tmp by 1 to sort next set
79     }
80     selSortR(arr, mid, length); //Call itself
81 }
82
83 }
84
85 }
```

Note: Full testing code, and main function on .c file.